BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan oleh penulis pada data *tweet* tentang wisata di Yogyakarta pada bulan November 2016 hingga Desember 2016 dapat diambil kesimpulan bahwa dengan Metode Apriori dapat digunakan dalam menentukan paket wisata di Kota Yogyakarta meskipun menggunakan *minimum support* kecil data dapat digunakan dan menghasilkan pilihan paket-paket wisata.

5.2. Saran

Adapun beberapa saran yang disampaikan untuk penentuan paket wisata di Kota Yogyakarta adalah sebagai berikut :

1. Dari hasil penelitian ini dapat digunakan untuk Tukang Becak atau Delman dalam menawarkan paket wisata kepada wisatawan yang ada di Kota Yogyakarta.
2. Dari hasil penelitian ini banyak terdapat data *tweets* yang tidak sesuai dengan harapan penulis karena data yang digunakan terlalu umum seperti wisata jogja, disarankan untuk penulis berikutnya untuk menggunakan kata kunci lebih spesifik seperti Malioboro, Tugu Yogyakarta.

DAFTAR PUSTAKA

Azizurahman, S. (2011), Analisis Dan Implementasi Metode N-Gram Pada Information Retrieval, Tugas Akhir, Fakultas Teknik Informatika Universitas Telkom.

Badri Muhmmad,(2012),*Social Media reltaion di Era Web*, Jurnal Risalah Vol. XXI, Edisi April.

Desky. (2001). *Manajemen Perjalanan Wisata*. Yogyakarta: Adicipta Karya Nusantara.

Fauzi,A.(2009). *All About Twitter*. Bandung: Mizan Media Utama.

Fürnkranz,J., (2009),"*A Study Using N-gram Features for Text Categorization*," Austrian Research Institute for Artificial Intelligenc.

Han, J, Kamber, M,& Pei, J.(2006). *Data Mining: Concept and Techniques*, Second Edition. Waltham: Morgan Kaufmann Publishers.

Han,Jiawei dan Kamber, Micheline,(2006),*Data Mining: Concept and Techniques Second Edition*, Morgan Kaufmann Publishers.

Kominfo. (2013). *Pengguna Internet di Indonesia 63 Juta Orang*, Jakarta.

Kusrini dan Emha Taufik Luthfi, 2009, *Algoritma Data Mining*, Yogyakarta: Penerbit Andi.

Larose D, T., 2005, *Discovering knowledge in data : an introduction to data mining*, Jhon Wiley & Sons Inc.

Lee, Ickjai, Cai, Guochen, dan Lee, Kyungmi, 2013, *Mining Points-of-Interest Association Rules from Geo-tagged Photos*, 46th Hawaii International Conference on System Sciences.

Listriani, D. (2016), Penerapan Metode Asosiasi Menggunakan Algoritma Apriori Pada Aplikasi Analisa Pola Belanja Konsumen (Studi Kasus Toko

Buku Gramedia Bintaro), Jurnal Teknik Informatika Vol 9 No. 2, Oktober 2016.

Lynch, K. (1960). *The Image Of The City*. Cambridge, Massachusetts, and London: The M.I.T. Press.

Pertiwi, L.A. (2012), Implementasi Algoritma Apriori Dalam Pembuatan Perjalanan Paket Wisata Ke Propinsi Daerah Istimewa Yogyakarta. Skripsi. Fakultas Teknologi Informatika Universitas Kristen Satya Wacana Salatiga.

Pramudiono, I., 2007, Algoritma Apriori, http://datamining.japati.net/cgi-bin/indodm.cgi?bacaarsip&117221014.

Ridwan,M.(2012),*Perencanaan dan Pengembangan Pariwisata*. Medan:PT SOFMEDIA.

Santa Ana Planning Division. (2010). *City of Santa Anna General Plan : Urban Design Element 1998.* Santa Ana: City Of Santa Ana Planning Division.

Sujali, (2011). *Geografi Pariwisata dan Kepariwisataan*. Yogyakarta: Fakultas Geografi UGM.

Turban, E., dkk. 2005. *Decision Support Systems and Intelligent Systems*. Yogyakarta: Andi Offset.

Ujang, N., & Kum, T. L. (2014). Identification of *Nodes* as Legible Features in the Historical District of Kuala Lumpur. *Australian Journal of Basic and Applied Sciences*, 96-105.

Ulmer, David, (2002), "*Mining an Online Auctions Data Warehouse*." The Mid-Atlantic Student Workshop on Programming Languagesand Systems. Pace University. http://csis.pace.edu/csis/masplas/p8.pdf.

Yoeti, A. Oka. (1997). *Perencanaan dan Pengembangan Pariwisata*. Jakarta: PT Pradnya Paramita.

LAMPIRAN

Lampiran 1 – Pseudocode

1. Kode Program untuk Data Collection dengan bahasa Java

```
package me.jhenrique.main;

import control.Control;
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.List;
import me.jhenrique.manager.TweetManager;
import me.jhenrique.manager.TwitterCriteria;
import me.jhenrique.model.Tweet;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;
import javax.management.Query;
public class Main {
    private static Connection connection;
    public static String url ="jdbc:ucanaccess://";
    public           static          final           String
path="E:"+File.separator+"Malioboro.mdb";

 private static final String USERNAME = "Username: ";
 private static final String RETWEETS = "Retweets: ";
 private static final String TEXT = "Text: ";
    private static final String DATE ="Date: ";
 private static final String MENTIONS = "Mentions: ";
 private static final String HASHTAGS = "Hashtags: ";
       private static final String LOCATION ="Location: ";

 public static void main(String[] args) throws SQLException,
ClassNotFoundException {
  /**
   * Reusable objects
   */
     Control c= new Control();
                  try {
            connection                                          =
DriverManager.getConnection(url+path);
            System.out.println("sukses");
        } catch (SQLException ex) {

Logger.getLogger(Main.class.getName()).log(Level.SEVERE,
null, ex);
        }
  TwitterCriteria criteria = null;
  Tweet t = null;
```

```
do{
 criteria = TwitterCriteria.create()

.setQuerySearch("wisata jogja")
.setSince("2016-11-01")
.setUntil("2016-12-10");

List<Tweet> tweets = TweetManager.getTweets(criteria);
int i =1;
for ( Tweet tw : tweets)
                        {
System.out.println(i);
System.out.println("### Example 2 - Get tweets by query
search");
System.out.println(USERNAME + tw.getUsername());
System.out.println(RETWEETS + tw.getRetweets());
System.out.println(TEXT + tw.getText());
System.out.println(DATE + tw.getDate());
System.out.println(MENTIONS + tw.getMentions());
System.out.println(HASHTAGS + tw.getHashtags());
System.out.println(LOCATION + tw.getGeo());
System.out.println();
i=i+1;

String sql="INSERT INTO BI(username, Status, tanggal,
retweet, favorites, mentions, hastags, geo_location)
VALUES(?,?,?,?,?,?,?,?)";

                                PreparedStatement pStmt =
connection.prepareStatement(sql);
                          //                  pStmt.setInt(1,
c.GetRowDataTwitter());
pStmt.setString(1, tw.getUsername().toString());
pStmt.setString(2, tw.getText());
pStmt.setString(3, tw.getDate().toString());
pStmt.setInt(4, tw.getRetweets());
pStmt.setInt(5, tw.getFavorites());
pStmt.setString(6, tw.getMentions());
 pStmt.setString(7,tw.getHashtags());
pStmt.setString(8,tw.getGeo().toString());
 pStmt.executeUpdate();
    }
  break;

}while(TweetManager.getTweets(criteria).size()!=0);
 }
```

2. Kode Program Stopwords

```
package com.uttesh.exude.stemming;

import com.uttesh.exude.ExudeData;
import com.uttesh.exude.exception.InvalidDataException;
import static com.uttesh.exude.stemming.Stemmer.c;
import static com.uttesh.exude.stemming.Stemmer.path;
import static com.uttesh.exude.stemming.Stemmer.url;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

/**
 *
 * @author Andjar
 */
public class Stopwords {

    /**
     * @param args the command line arguments
     */
    public   static   void   main(String[]   args)   throws
InvalidDataException {
        // TODO code application logic here

        String sql = "SELECT * FROM Twitter ";
        System.out.println("Sedang diproses....");

        try
        {
            c=DriverManager.getConnection(url+path);
            System.out.println("Berhasil konekk database");
            Statement state = c.createStatement();
            ResultSet rs = state.executeQuery(sql);
            if(rs!=null)
            {
                while(rs.next())
                {
                    int id = rs.getInt("ID");
String username= rs.getString("username");
String inputData = rs.getString("Status");
String tanggal = rs.getString("tanggal");
int retweet = rs.getInt("retweet");
int favorites=rs.getInt("favorites");
String mention = rs.getString("mentions");
String hastags =rs.getString("hastags");
String geo_location=rs.getString("geo_location");
```

```
String                          output                          =
ExudeData.getInstance().filterStoppingsKeepDuplicates(inputD
ata);
//String sql2="UPDATE Stopwords set Status = ? where ID =?";
//versi edit
String      sql2      =      "INSERT      into      stopwords
values(?,?,?,?,?,?,?,?,?)"; //versi insert
//System.out.println("output ID "+id+" : "+output);
PreparedStatement pStmt = c.prepareStatement(sql2);
// pStmt.setInt(1, c.GetRowDataTwitter());
                        pStmt.setInt(1, id);
                        pStmt.setString(2, username);
                        pStmt.setString(3, output);
                        pStmt.setString(4, tanggal);
                        pStmt.setInt(5, retweet);
                        pStmt.setInt(6, favorites);
                        pStmt.setString(7, mention);
                        pStmt.setString(8, hastags);
                        pStmt.setString(9, geo_location);

                        pStmt.executeUpdate();
                    }
                }
                rs.close();
                state.close();
                 c.close();
                 System.out.println("database ditutup");
            }
            catch(Exception EX)
            {
                System.out.println("Error Reading From database.
. .");
                System.out.println(EX);
            }
 }
}
```

3. Kode Program N-Gram untuk cari tempat wisata yang ada

```
package n.gram;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
import java.util.function.Function;
import java.util.stream.Collectors;
import static java.util.stream.Collectors.counting;
import java.util.stream.Stream;
```

```
/**
 *
 * @author andjar
 */
public class BagiKata {
 public static List<String> ngrams( String str) {
     int n=1;
        List<String> ngrams = new ArrayList<String>();
        String[] words = str.split(" ");

        for (int i = 0; i < words.length - n + 1; i++)
        {

if(words[i].equalsIgnoreCase("tugu")||words[i].equalsIgnoreC
ase("museum")||words[i].equalsIgnoreCase("benteng")||words[i
].equalsIgnoreCase("taman”))
            {
                n=2;
                ngrams.add(concat(words, i, i+n));
                i++;
            }
            else
            {
                n=1;
                ngrams.add(concat(words, i, i+n));
            }
        }
        return ngrams;
    }

    public static String concat(String[] words, int start,
int end) {
        StringBuilder sb = new StringBuilder();
        for (int i = start; i < end; i++)
            sb.append((i > start ? " " : "") + words[i]);
        return sb.toString();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws
IOException {
        // TODO code application logic here
        //for (int n = 1; n <= 2; n++) {
        String temp="";
        String filePath = "E:/testing.txt"; // letak file
yang akan di pecah per kata

        for (String ngram : ngrams(readLineByLineJava8(
filePath )))
```

```
            {
                System.out.println(ngram);
            }

    }
    private    static    String    readLineByLineJava8(String
filePath)
    {
        StringBuilder contentBuilder = new StringBuilder();
        try    (Stream<String>    stream    =    Files.lines(
Paths.get(filePath), StandardCharsets.UTF_8))
        {
           stream.forEach(s                                    ->
contentBuilder.append(s).append("\n"));
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        return contentBuilder.toString();
    }
        static    void    countWords(String    word)    throws
IOException {
        Arrays.stream(word.split("[\\r\\n]+"))

.collect(Collectors.groupingBy(Function.<String>identity(),
TreeMap::new, counting())).entrySet()
        .forEach(System.out::println);
    }
}
```

Lampiran 2 – Hasil Run WEKA 3.81

```
=== Run information ===

Scheme:       weka.associations.Apriori -I -R -N 100 -T 0 -C
0.5 -D 0.05 -U 1.0 -M 0.01 -S -1.0 -c -1
Relation:     TABEL OLAH ALL-
weka.filters.unsupervised.attribute.Remove-R1-2,11-
13,18,20,25
Instances:    188
Attributes:   18
              Malioboro
              Tugu
              Alun-alun utara
              alun-alun selatan
              taman sari
              benteng vredeburg
              gembira loka
              sindu kesuma
              Stasiun Lempuyangan
              Stasiun Tugu
              Keraton
              BNI 46 Nol Kilometer
              Kotabaru
              Gedung Agung
              Kauman
              Pasar Beringharjo
              Taman Budaya Yogyakarta
              Masjid Agung
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.01 (2 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 20

Generated sets of large itemsets:

Size of set of large itemsets L(1): 14

Large Itemsets L(1):
Malioboro=Y 84
Tugu=Y 52
Alun-alun utara=Y 46
alun-alun selatan=Y 27
taman sari=Y 36
benteng vredeburg=Y 18
gembira loka=Y 9
Stasiun Lempuyangan=Y 6
Stasiun Tugu=Y 25
```

```
Keraton=Y 53
BNI 46 Nol Kilometer=Y 38
Kauman=Y 2
Pasar Beringharjo=Y 9
Taman Budaya Yogyakarta=Y 4

Size of set of large itemsets L(2): 36

Large Itemsets L(2):
Malioboro=Y Tugu=Y 27
Malioboro=Y Alun-alun utara=Y 13
Malioboro=Y alun-alun selatan=Y 14
Malioboro=Y taman sari=Y 8
Malioboro=Y benteng vredeburg=Y 9
Malioboro=Y gembira loka=Y 3
Malioboro=Y Stasiun Tugu=Y 7
Malioboro=Y Keraton=Y 16
Malioboro=Y BNI 46 Nol Kilometer=Y 8
Malioboro=Y Pasar Beringharjo=Y 5
Tugu=Y Alun-alun utara=Y 6
Tugu=Y alun-alun selatan=Y 3
Tugu=Y taman sari=Y 7
Tugu=Y benteng vredeburg=Y 2
Tugu=Y Stasiun Tugu=Y 4
Tugu=Y Keraton=Y 8
Tugu=Y BNI 46 Nol Kilometer=Y 9
Alun-alun utara=Y alun-alun selatan=Y 6
Alun-alun utara=Y taman sari=Y 6
Alun-alun utara=Y benteng vredeburg=Y 3
Alun-alun utara=Y Stasiun Tugu=Y 3
Alun-alun utara=Y Keraton=Y 15
Alun-alun utara=Y BNI 46 Nol Kilometer=Y 9
alun-alun selatan=Y taman sari=Y 7
alun-alun selatan=Y Stasiun Tugu=Y 3
alun-alun selatan=Y Keraton=Y 6
taman sari=Y Stasiun Tugu=Y 2
taman sari=Y Keraton=Y 17
taman sari=Y BNI 46 Nol Kilometer=Y 3
benteng vredeburg=Y Keraton=Y 3
benteng vredeburg=Y BNI 46 Nol Kilometer=Y 8
gembira loka=Y Keraton=Y 3
Stasiun Lempuyangan=Y Stasiun Tugu=Y 2
Stasiun Tugu=Y BNI 46 Nol Kilometer=Y 3
Keraton=Y BNI 46 Nol Kilometer=Y 2
BNI 46 Nol Kilometer=Y Taman Budaya Yogyakarta=Y 2

Size of set of large itemsets L(3): 15

Large Itemsets L(3):
Malioboro=Y Tugu=Y Alun-alun utara=Y 2
Malioboro=Y Tugu=Y alun-alun selatan=Y 2
```

```
Malioboro=Y Tugu=Y benteng vredeburg=Y 2
Malioboro=Y Tugu=Y Keraton=Y 4
Malioboro=Y Tugu=Y BNI 46 Nol Kilometer=Y 3
Malioboro=Y Alun-alun utara=Y alun-alun selatan=Y 3
Malioboro=Y Alun-alun utara=Y taman sari=Y 3
Malioboro=Y Alun-alun utara=Y benteng vredeburg=Y 2
Malioboro=Y Alun-alun utara=Y Keraton=Y 3
Malioboro=Y alun-alun selatan=Y taman sari=Y 3
Malioboro=Y alun-alun selatan=Y Keraton=Y 2
Malioboro=Y taman sari=Y Keraton=Y 2
Alun-alun utara=Y alun-alun selatan=Y taman sari=Y 3
Alun-alun utara=Y alun-alun selatan=Y Keraton=Y 2
alun-alun selatan=Y taman sari=Y Keraton=Y 2

Size of set of large itemsets L(4): 1

Large Itemsets L(4):
Malioboro=Y Alun-alun utara=Y alun-alun selatan=Y taman
sari=Y 2

Best rules found:

  1. Tugu=Y benteng vredeburg=Y 2 ==> Malioboro=Y 2
<conf:(1)> lift:(2.24) lev:(0.01) [1] conv:(1.11)
  2. Tugu=Y alun-alun selatan=Y 3 ==> Malioboro=Y 2
<conf:(0.67)> lift:(1.49) lev:(0) [0] conv:(0.83)
  3. Alun-alun utara=Y benteng vredeburg=Y 3 ==> Malioboro=Y
2    <conf:(0.67)> lift:(1.49) lev:(0) [0] conv:(0.83)
  4. Alun-alun utara=Y alun-alun selatan=Y taman sari=Y 3
==> Malioboro=Y 2    <conf:(0.67)> lift:(1.49) lev:(0) [0]
conv:(0.83)
  5. Malioboro=Y alun-alun selatan=Y taman sari=Y 3 ==>
Alun-alun utara=Y 2    <conf:(0.67)> lift:(2.72) lev:(0.01)
[1] conv:(1.13)
  6. Malioboro=Y Alun-alun utara=Y taman sari=Y 3 ==> alun-
alun selatan=Y 2    <conf:(0.67)> lift:(4.64) lev:(0.01) [1]
conv:(1.28)
  7. Malioboro=Y Alun-alun utara=Y alun-alun selatan=Y 3 ==>
taman sari=Y 2    <conf:(0.67)> lift:(3.48) lev:(0.01) [1]
conv:(1.21)
  8. Pasar Beringharjo=Y 9 ==> Malioboro=Y 5
<conf:(0.56)> lift:(1.24) lev:(0.01) [0] conv:(1)
  9. Tugu=Y 52 ==> Malioboro=Y 27    <conf:(0.52)>
lift:(1.16) lev:(0.02) [3] conv:(1.11)
 10. alun-alun selatan=Y 27 ==> Malioboro=Y 14
<conf:(0.52)> lift:(1.16) lev:(0.01) [1] conv:(1.07)
 11. benteng vredeburg=Y 18 ==> Malioboro=Y 9
<conf:(0.5)> lift:(1.12) lev:(0.01) [0] conv:(1)
 12. Tugu=Y Keraton=Y 8 ==> Malioboro=Y 4    <conf:(0.5)>
lift:(1.12) lev:(0) [0] conv:(0.89)
```

13. Alun-alun utara=Y alun-alun selatan=Y 6 ==> Malioboro=Y 3 <conf:(0.5)> lift:(1.12) lev:(0) [0] conv:(0.83)
14. Alun-alun utara=Y taman sari=Y 6 ==> Malioboro=Y 3 <conf:(0.5)> lift:(1.12) lev:(0) [0] conv:(0.83)
15. Alun-alun utara=Y taman sari=Y 6 ==> alun-alun selatan=Y 3 <conf:(0.5)> lift:(3.48) lev:(0.01) [2] conv:(1.28)
16. Alun-alun utara=Y alun-alun selatan=Y 6 ==> taman sari=Y 3 <conf:(0.5)> lift:(2.61) lev:(0.01) [1] conv:(1.21)
17. Taman Budaya Yogyakarta=Y 4 ==> BNI 46 Nol Kilometer=Y 2 <conf:(0.5)> lift:(2.47) lev:(0.01) [1] conv:(1.06)